Integrated Tools and Techniques Applied to the TES Ground Data System[1]

Brian A. Morrison
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109
818-354-2458
Brian.Morrison@jpl.nasa.gov

Abstract—The Tropospheric Emission Spectrometer (TES) is a Fourier Transform Spectrometer (FTS) scheduled for launch in June of 2003. Over its six-year mission, it is expected to provide three-dimensional maps of ozone and its precursors as part of NASA's Earth Observing System (EOS). Developing production-quality software to process large volumes of scientific data and accommodate changing requirements throughout the mission along with the need to ensure that scientific goals and objectives are met presents many challenges. The author of the paper will discuss the selection of CASE tools, a decision making process, requirements tracking and a review mechanism that leads to a highly integrated approach to software development that must deal with the constant pressure to change software requirements and design that is associated with research and development. This integrated approach of tools and techniques should provide a process and mechanism to assist in continually verifying that mission objectives are correctly expressed in software requirements and fully realized in design in order to achieve overall mission success of the ground data processing system.

TABLE OF CONTENTS

1. INTRODUCTION

TES Mission

TES is one of a series of polar-orbiting satellites in NASA's Earth Observing System (EOS), which is an effort to obtain long-term global observations of the land surfaces, the biosphere, solid earth, the atmosphere and oceans. EOS itself is the principal element of an even larger venture called Earth Science Enterprise (ESE). This mission is a concerted effort on the part of NASA and the international community to understand our planet's climate system and its variations.

TES has two basic science operating modes: Global Surveys (GS) and Special Research Observations. For Global Surveys, continuous sequences of a space view and a blackbody view calibration pair, two nadir views and 3 limb views are acquired. Calibrations and nadir views require 4 seconds each, limb views 16 seconds. Adding in the times needed for accelerating and decelerating the moving element of the FTS, each sequence requires 81.2 seconds to accomplish. 73 sequences are acquired on each orbit, triggered by passage of the orbital southern apex, and an entire survey requires 58 orbits (just under 4 days). Each survey is preceded and followed by 2 orbits of pure space and blackbody views for calibration purposes. The CHEM orbit has a 16-day repeat period, so Global Surveys are made on a "4-day-on, 4-day-off" cycle.

Special Research Observations fall into two general categories. The first category is to collect targeted nadir observations of specific locations such as volcanoes or biomass burning. Such observations are made for as long as the target is within ±45E of the nadir direction (up to 210 seconds). The second category is to make transect observations: up to about 800 km long down-looking and essentially indefinitely at the limb. In every case, such observations are accompanied by appropriate calibration sequences. In general, Special Research Observations are made during the 4-day gaps in the Global Surveys [1].

2. GROUND DATA SYSTEM

Of the two modes, GS is the workhorse mode for TES whereby the majority of data will be collected and processed in order to produce standard data products that will be distributed to the scientific community on a regular basis. This collection and processing of both GS and Special Observations of TES instrument data is expected to last approximately five years.

The very nature of TES with its voluminous data

[1] 0-7803-6599-2/01/$10.00 © 2001 IEEE

computational requirements along with newly developed algorithms present several challenges to the developers of the science software needed to process the data.

Processing Levels

The TES Ground Data Processing System is comprised of four major subsystems, Level-1A, Level-1B, Level-2 and Level-3. Each subsystem is a fairly large transformational step needed to further process TES data in order to produce the final data products.

Level-1A

Level-1A converts the serial bit stream from the spacecraft back to interferograms (i.e., essentially the original instrument output). The signal amplitudes at this stage are in Data Numbers (DN), which can range between $\pm(2^{15}-1)$, on an (implicit) grid of Optical Path Difference (OPD) determined by the frequency of the Nd:YAG on-board control laser and the number of laser fringes between samples (specified in Table XIX of the SOAGR). Level- A is largely non-algorithmic with the exception of the computations of pointing angles and target locations based on shaft encoder outputs from the Pointing Control System (PCS), spacecraft attitude (from the on-board gyro) and spacecraft position (from the ephemeris). Short-scan (4 sec) interferograms contain between about 15,000 and 20,000 points. Long-scan (16 sec) interferograms are 4 times larger. All Level-1A processing occurs at the SIPS.

Level-1B

Level-1B converts the interferograms, through a process of phase correction and calibration, to radiometrically-calibrated spectra. The output amplitudes at this stage are in watts/cm^2/sr/cm^{-1} on a grid of absolute vacuum cm^{-1}. While quite computationally-intensive, Level-1B is much less so than Level-2 (estimated to be about 10% of the Level-2 loading). All Level-1B processing occurs at the SIPS.

Level-2

This level converts the spectra to vertical volume mixing ratios of selected molecules on a pre-determined pressure (not altitude) grid. This process, called *Earth Limb and Nadir Operational Retrieval* (ELANOR), is the most computationally-intensive part of the entire data processing and therefore receives the most attention during development in order to find ways of reducing the processing burden. The volumes of the retrieved profiles are very small. However, a final stage in the process is to compute a complete spectrum based on the retrieved profiles and subtract it from the observed Level-1B spectrum. This file of *residuals* is the same size as Levels-1A and 1B outputs. Standard Product Level-2 processing occurs at the Science Investigator-Led Processing System (SIPS); Special Research Products Level-2 processing occurs at the Science Computing Facility (SCF) located at the Jet Propulsion Laboratory.

Level-3

This level generates global maps of species on selected pressure or isoentropic surfaces to facilitate browsing and comparison to models and other observations (spaceborne or *in situ*). This process is usually undertaken only for Global Surveys and occurs at the SIPS, although we shall probably wish to use a simplified version at the SCF for transects.

3. CHANGE IS THE CHALLENGE

Several challenges within the software development effort must be dealt with to ensure that overall success is achieved. The challenges that the TES team face are: a large and complex software system, a system development that occurs over several years, continually changing requirements, a need for up to date documentation, constraints on resources (including personnel and budget) and most importantly the need to ensure that the software produced meets the scientific goals and objectives set forth by the TES science team.

The Problem

So the problem that the TES software development team faces is how to capture and implement requirements of a large and complex science software system developed over several years that must also deal with constant algorithmic change (requirements modifications) through development and after deployment while still trying to achieve the goals and objectives set forth by the science team.

So the problem becomes more than the careful selection of vertical tools such as compilers, editors, debuggers, etc. While these items remain an important element in any software development effort it is not enough to guarantee that the steps within the software development lifecycle are consistent with the overall goals and objectives. It has become apparent to the author that there is a critical need for horizontal tools (requirements tracking, unified design modeling and uniform document generation) along with formal peer reviews and a high level of interactive collaboration between scientist and software developer to assist in attaining mission success. Unlike the development and deployment of communication satellites that typically have stable functional requirements that when delivered either work or don't work (with the exception of occasional software patches that are uplinked to fix bugs or provide some limited additional functionality), the TES instrument is predominately research and development in nature.

A Need for Process and CASE Tools

It is the authors opinion that in order for the TES software development team to be successful in a changing environment there must be a well-defined process for managing change coupled with the appropriate choice of horizontal CASE tools to aide in the managing, tracking and deployment of science software.

5. Integrated Development Approach

On the TES project, the software team has selected an integrated approach that combines both a human processes for establishing, managing and implementing requirements and specific horizontal CASE tools for managing requirements, centralizing a design, generating documentation, tracking action items and controlling changes. With this approach, scientist and software developers work closely together to address issues and manage change.

As discussed here, an integrated approach means addressing the need to have both human processes and processes that can be furthered by the used of CASE tools and most importantly a useful connection between both types of processes. Figure 1 shows how the two processes are related and form an integrated approach to managing the lifecycle. It may be tempting in software development in the face of great change to ignore tracking and managing of that change, after all if it is always changing why even bother recording it? Often this approach is adopted for rapid development or proof-of-concept development efforts, which may be appropriate. But for large production quality projects that are not prototypes this approach beckons failure. It is the author's opinion that it is the very nature of change that can lead development astray by letting the seemingly small incremental modifications (also known as requirements creep) to slowly move the entire effort off-track. Therefore, we must manage the change. Next we discuss the human processes in managing change.

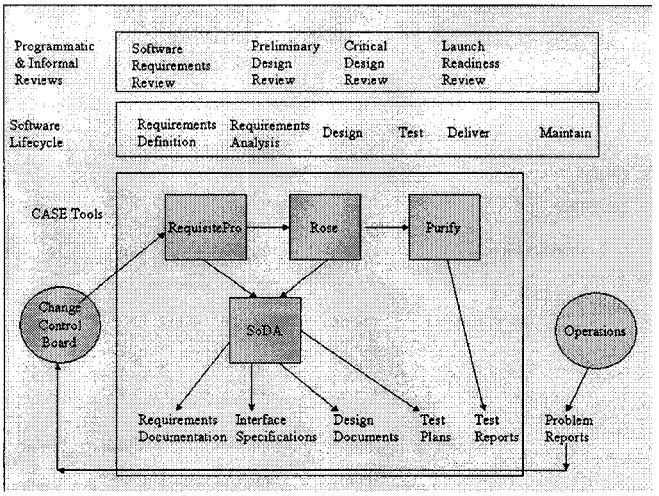

Figure 1 - Integrated CASE Tools

4. Teaming To Understand Requirements

The science team and the software development team meet formally once a week, and informally daily. It is through these frequent sessions that the ambiguity of requirements are reduced or completely eliminated altogether. So the process is not one of gained requirement content simply by the transcription of information gathered and written into itemized requirements. Rather, it is the process of frequent interaction using diagrams, drawings and words to express a science requirement in ways comfortable to both parties that help better define requirements [4].

Software Peer Reviews

On the TES project we use the peer review process as one technique to better understand science requirements. This technique is familiar ground to the scientist and represents a fairly common mechanism to attempt to better understand science goals and objectives by emphasizing what is known and to expose areas of weak understanding. The key to developing requirements during the peer review process is to understand that the science requirements themselves are evolving in parallel to the scientific research. Through peer reviews we embed the development of requirements with the process of advancing the research by working collectively (scientist and software developer) to better understand the goals and objectives of the science.

Example of Change by Review

For example, on TES there is a high level requirement "to have a capability to visualize science software results." This requirement originates at a high level requirement levied by the Principal Investigator (PI), which traces down to a system level requirement, and then ultimately down to a visualization subsystem requirement. Through the process of an internal review of the visualization software requirements, the board discovered a derived requirement that stipulated that the visualization software needs to be "highly reliable" and is "not mission critical". The science members on the review board, however, indicated that this may not be true and determined it needed further attention. This finding generated a review action item. The science team determined that the science results gathered from the use of visualization tools may be used to make major changes in algorithms or instrument operations and was therefore a critical element in overall mission success. The item was worked offline and the requirement was subsequently changed to indicate that the software is "mission critical". The change to mission critical status indicates an important step in the requirements development process. A derived requirement was reviewed by the customer (science team), and corrective changes were made early that greatly affect mission success.

So the goal then is not only to capture requirements more accurately, but to capture requirements as they are presented, clarified, reworked and understood by the scientist and engineer in a social context. It is this highly interactive exchange of refining and understanding the meaning behind the requirements that the author believe gives the greatest value to evolved requirements on the TES project.

Change Control Board (CCB)

As is normal for a formal project, all changes to algorithms, codes, documentation and processes are subject to change control. Once software requirements have been baselined any change must go through the Change Control Board (CCB). The CCB is chaired by the PI and meets as frequently as necessary to ensure expeditious action on change requests.

The usage of the CCB comprised of both scientist and software developers provides an excellent means for controlling change. Changes to requirements such as priorities, testing methods, resource allocation and decisions of deletion and addition of requirements are made regularly with a conscientious view as to how the changes will impact overall science mission objectives. In this manner, requirements and their associated attributes can be changed without jeopardizing the mission by losing sight of the bigger picture.

6. HORIZONTAL CASE TOOLS

The second element of an integrated approach to managing and controlling change is accomplished through the use of CASE tools, specifically horizontal CASE tools. Horizontal CASE tools are tools that span across the software development lifecycle. Vertical CASE tools are targeted usually within a specific stage of development. Examples of vertical tools include compiler, editors, run-time checkers etc. Therefore, for purposes of managing and controlling change across the development lifecycle, mostly horizontal tools will be discussed.

Using Rational RequisitePro To Manage Requirements

On the TES project we, currently use RequisitePro, a software requirements management and tracking tool provided by Rational Software, Inc. The use of such a tool may seem obvious. There are many stories of failed software development efforts that can be traced back to a lack of fully understanding and capturing initial requirements coupled with problems associated with not managing changing requirements throughout the software development lifecycle [2].

We use RequisitePro to capture requirements, add needed project specific attributes to requirements, trace requirements to other higher level requirements or algorithms, build test plans and map requirements to programmatic deliveries. These are all part of the many steps necessary to developing successful software. We agree that such a tool is an excellent way of managing and tracking change of software requirements. However, for our team the true value extends beyond the benefits of merely bookkeeping requirements. Figure 2 shows how RequisitePro has been tailored to incorporate requirement attributes specific to our project.

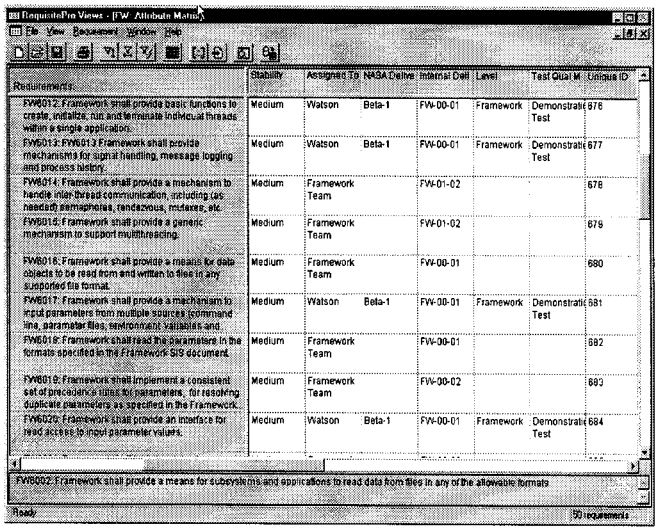

Figure 2 - Customized Requirement Attributes using ReqPro

We are hopeful the tool will become a method for furthering the communication between scientist and requirements authors. This is accomplished by using the tool as a way to continually refine science requirements (a process that is ongoing irrespective of software development). In addition, we have extended the attributes within the tool to incorporate project specific details: subsystem names, delivery phases, test methods, names of developers, science priorities, and so on. In this way, the scientists see the requirements as more than just a lengthy list of itemized unrelated facts. The tool also provides an easy-to-use interface (RequisitePro is accessed through Microsoft Word) and, finally, the tool provides easy access to the requirements, again furthering the idea that the scientist may directly add new requirements, annotate existing ones or simply view them [3]. Figure 3 shows the flow of requirements change using RequisitePro.

The tool therefore is not viewed as a peripheral activity; instead it has become an integrated part to the already existing collection of tools used in the scientific research process. This collaborative use of the tool has contributed to a better understanding of the science requirements by viewing science requirement development as a highly integrated piece in the effort to establish specific research

goals.

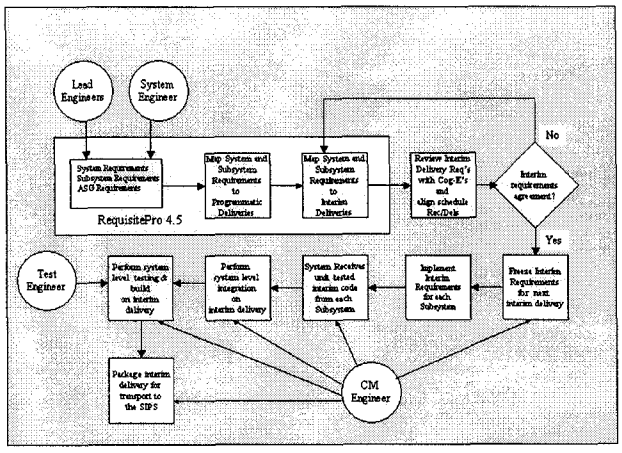

Figure 3- Flow of requirements change.

Using Rational Rose To Centralize a Design

Rational Rose is a visual modeling tool provided by Rational Software Inc. that is based on the Unified Modeling Language™ (UML). The UML is currently the standard notation for object-oriented software architecture. Rational Rose is used by our team to visualize, understand, and refine our requirements (tracked and managed by RequisitePro) and architecture before committing them to code. This allows the team to avoid wasted effort in the development cycle. By using a single modeling tool throughout the development life cycle it should help us ensure that we are building the right system. The architecture model can be traced back to the system requirements, although currently this step is not handled in an automated fashion.

The entire design resides in one model that is accessible by all team members. Changes and modifications to the design model are controlled by Harvest/CCC, the configuration management tool used on TES. Model files are checked out, modified and checked back into Harvest in the same manner as code, design file memorandums and other supporting documentation. Using Rational Rose we are able to centralize the entire design for the SDPS and have change control with versioning.

Since the Rational Rose design is expressed using a visual modeling tool, we have been able to visually communicate the design to the TES science team without them having to understand many of the underlying details. The science team has made a concerted effort to understand and provide feedback about various design elements using UML. This greatly facilitates communication of how requirements are implemented in design and provides valuable feedback to the developers.

Rational Rose provides the capability to generate C++ code which is planned for TES science software. We are hopeful that we will be able to use Rose and CCC/Harvest together in order to implement round-trip engineering. With round-trip engineering, a delivered version of software code could be checked out from Harvest, loaded into Rose and then appropriate design modifications would only occur within the visual model. New C++ code would then be generated, tested, checked into Harvest as a newer version and then ultimately delivered to the target run-time environment.

Using Rational SoDA to Generate Documentation

Rational SoDA is a document tool provided by Rational Software, Inc. Like most large software development efforts, communicating project information can be time consuming and difficult. Project lifecycle documentation and reports such as requirements, interfaces and test plans are often produced haphazardly, or are neglected altogether because of the effort and time involved in creating and maintaining them. By using Rational SoDA we are attempting to overcome many of these issues by automating the creation and maintenance of comprehensive project documentation and reports with a single tool.

Unlike manual methods, SoDA generates complete documentation by automatically extracting data from various project tool databases. Currently on TES we use SoDA to generate interface documents by extracting attributes of selected classes within the Rose design model. In addition, we are using SoDA to generate a few requirement reports such as the entire list of all requirements and related traces extracted from the RequisitePro requirements database. In the future, we are planning to use SoDA to generate an external interface document describing our final data products. This document will be made available to the scientific community along with the data products. We are also planning on using Soda to generate test plans by extracting testable requirements from the RequisitePro database. In this way, test plans will be generated according to the most up-to-date requirements. Rational SoDa like Rational RequisitePro is built around Microsoft Word and is therefore fairly easy to use and generates documents that are in a format that can be read and edited by anyone using Word.

Using Rational Purify

Rational Purify is yet another tool provided by Rational Software Inc. Purify is a tool that helps locate run-time or memory-access errors. We use it on TES to automatically pinpoint hard-to-find run-time errors in our code and components. The report generated from Purify provides an important record in unit testing, integration testing and any regression testing that is performed. Purify although extremely useful on our team is really more of a vertical tool and is only mentioned here to describe the complete suite of Rational products used on the TES project.

7. CONNECTION - HUMAN PROCESSES AND TOOLS

On TES with an integrated approach, we attempt to bring together the need to have a well-defined process for managing change that deals with human processes and processes that can be furthered by use of software tools. The first involves establishing ways to have a closer working relationship with the science team. Daily interactions, peer reviews and a change control board have been discussed. The second establishes the implementation of several CASE tools to track and manage change as software is defined, designed and delivered. We have now discussed the various techniques and tools as ways to accomplish these two activities.

Integration By Involvement

Some of the integration of these two activities occurs by involving the science team members into the usage of the CASE tools. For example, by using Rational RequisitePro, scientists can view any requirement and also have direct input regarding specific requirements language and related attributes. Also, by exposure to Rational Rose, science team members have become very familiar with UML as the modeling language used to express the TES design.

Integration by Process

Integration of these two activities also occurs according to a workflow process. The process is not complicated. It is simply a planned path of both human processes and CASE tool usage and how the two are linked together through the entire software development lifecycle. This plan is currently being developed as part of the ongoing system engineering effort and is expected to be completed sometime in February of 2001.

8. CONCLUSION

We have addressed the problem that the TES software development team faces in how to capture and implement requirements of a large and complex science software system. We have shown that the software team has selected an integrated approach that combines both human processes for establishing, managing and implementing requirements along with specific horizontal CASE tools for managing requirements, centralizing a design, generating documentation and controlling changes. With this approach, scientist and software developers are working together to address issues and manage change. With this approach we are confident that the software team can work more easily and effectively with the science team.

It is hopeful that this integrated approach of tools and techniques will continue to provide a process and mechanism to assist in continually verifying that mission objectives are correctly expressed in software requirements and fully realized in design in order to achieve overall mission success.

ACKNOWLEDGEMENTS

The author would like to acknowledge Melinda C. Morrison, MBA for her valuable assistance in reviewing early drafts of this work. The work described herein was funded by the National Aeronautics and Space Administration (NASA) and performed at the Jet Propulsion Laboratory under the management of the California Institute of Technology.

REFERENCES

[1] Reinhard Beer, *TES Ground System Operations Concept*, D-18451, Caltech/JPL, CA, 2000.

[2] Richard H. Thayer and Merlin Dorfman, Eds. *Software Requirements Engineering*, 2nd Ed., Los Alamitos, CA: IEEE Computer Society Press, 1997.

[3] Humphrey, W.S., *Managing the Software Process*, Reading Mass, Addison Wesley, 1989.

[4] Steve Larson and Brian Morrison, Managing Software Requirements In the Context of the Scientific Enterprise, Pasadena, CA: IEEE Aerospace Conference, 2000.

Brian Morrison is a software system engineer at the Jet Propulsion Laboratory. He has been involved in the development of avionic software systems at Lockheed Aeronautical System and has worked on the command and control software for several NASA deep space tracking stations. Currently, he is involved in the development of a science data processing system as part of NASA's earth remote sensing program. Mr. Morrison has a B.S. in Computer Science from California State Polytechnic University and an MBA specializing in project development from the University of La Verne.